CSC 261/461
Database Systems

Eustrat Zhupa

November 12, 2018

Indexes

Index

- A database index is a data structure used to improve data access.
- There are two types of indices:
  1. Ordered indices: Based on a sorted ordering of the values.
  2. Hash indices: Based on a distribution of values across a range of buckets. The bucket is assigned by a function, called a hash function.

Indexing

Evaluation

- Indexing techniques are evaluated on the basis of:
    1. Access types: The types of access that are supported efficiently.
    2. Access time: The time it takes to find a particular data item.
    3. Insertion time: The time it takes to insert a new data item.
    4. Space overhead: The additional space occupied by an index structure.
- An attribute or set of attributes used to look up records in a file is called a `search key`.

Indexing

Index Structure

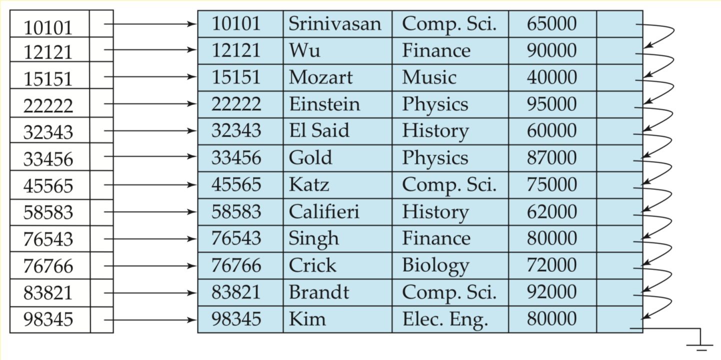
- an index structure can be used to gain fast random access to records in a file.
- each index structure is associated with a particular search key.
- an `ordered index` stores the search keys in sorted order, and associates with each search key the records that contain it.
- a `clustered index` follows the order of the rows on the disk in the same order as the index.
- `nonclustered` index specifies an order different from the sequential order of the file.

SQL

- An index entry, includes a search-key value and pointers to one or more records with that value as their search-key value.
- The pointer to a record consists of the identifier of a disk block and an offset within the disk block.
- There are two types of ordered indices:
  1. Dense index: an index entry appears for every search-key value in the file.
  2. Sparse index: an index entry appears for only some of the search-key values.

Indexing

Dense Index

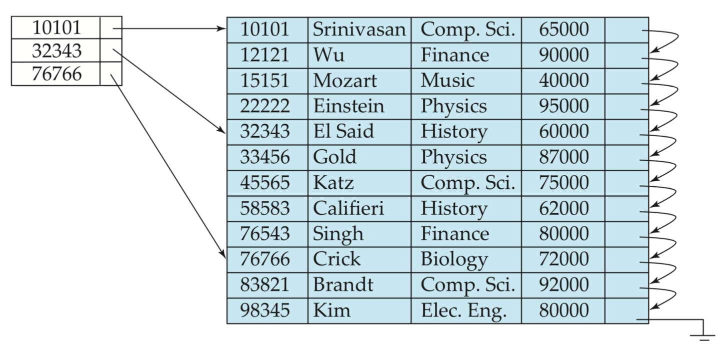


10101
12121
15151
22222
32343
33456
45565
58583
76543
76766
83821
98345

10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Indexing

Sparse Index

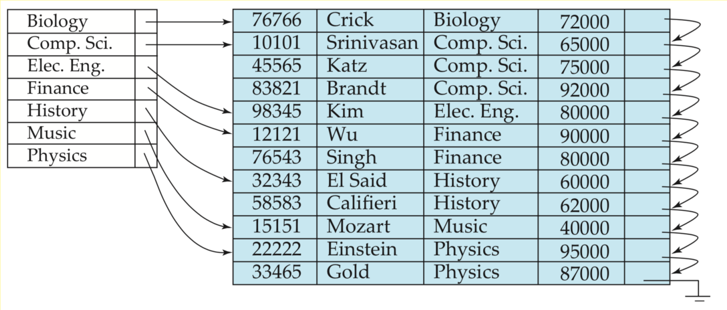


10101	
32343	
76766	

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Indexing

Dense Index



Biology
Comp. Sci.
Elec. Eng.
Finance
History
Music
Physics

76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
33465	Gold	Physics	87000

Indexing

Index Update

- index must be updated whenever a record is inserted/deleted
- in case a record is updated, affected index must be updated
- Insertion.
    - Dense indices:
        1. If value does not appear in the index, the system inserts an index entry at the appropriate position.
        2. Otherwise the following actions are taken:

           a. If the index entry stores pointers to all records with the same search-key value, add a pointer to the new record in the index entry.

           b. Otherwise, the index entry stores a pointer to only the first record with the search-key value. Add after existing records.
    - Sparse indices:
        1. If the system creates a new block, it inserts the first search-key value appearing in the new block into the index.
        2. if the new record has the least search-key value in its block, the system updates the index entry pointing to the block;

Indexing

Deletion

- Dense indices:
  1. If the deleted record was the only record with its value, delete index entry.
  2. Otherwise the following actions are taken:
     a. If the index entry stores pointers to all records with the same search- key value, the system deletes the pointer to the deleted record from the index entry.
     b. Otherwise, the index entry stores a pointer to first record. Updates the index entry to point to the next record.

Indexing

Deletion

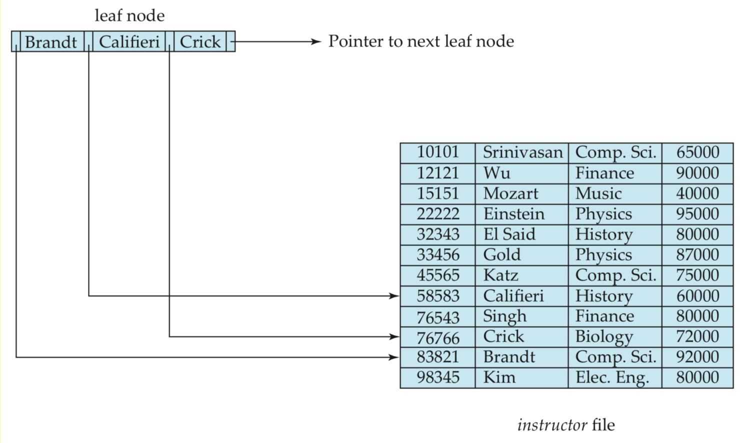
- Sparse indices:
  1. If the index does not contain an index entry with the search-key value of the deleted record, nothing needs to be done.
  2. Otherwise:

     a. If the deleted record was the only record with its search key, the system replaces index record with the next search-key value. If the next search-key value already has an index entry, the entry is deleted instead of being replaced.

     b. Otherwise, if the index entry for the search-key value points to the record being deleted, the system updates the index entry to point to the next record with the same search-key value.

Indexing

B+ Trees

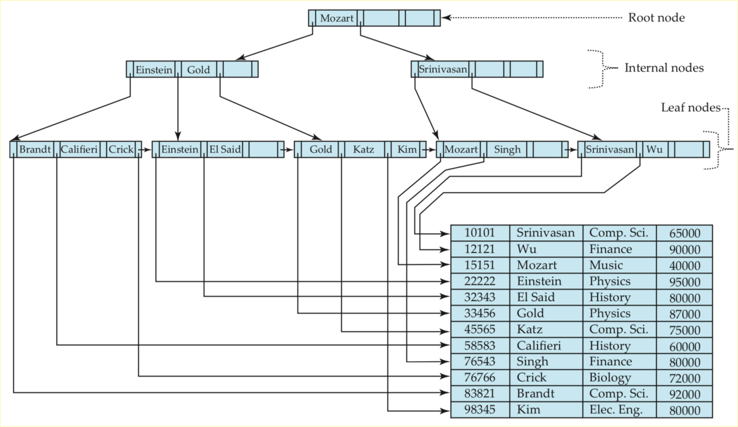


10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

instructor file

Indexing

B+ Trees



10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000